Firing Room Remote Application Software Development

Kan Liu[1]
NASA Kennedy Space Center, Merritt Island, FL 32899

The Engineering and Technology Directorate (NE) at National Aeronautics and Space Administration (NASA) Kennedy Space Center (KSC) is designing a new command and control system for the checkout and launch of Space Launch System (SLS) and future rockets. The purposes of the semester long internship as a remote application software developer include the design, development, integration, and verification of the software and hardware in the firing rooms, in particular with the Mobile Launcher (ML) Launch Accessories subsystem. In addition, a Conversion Fusion project was created to show specific approved checkout and launch engineering data for public-friendly display purposes.

Nomenclature

ACL	=	Application Control Language
ASPU	=	Aft Skirt Purge Umbilical
CAA	=	Crew Access Arm
CSFSU	=	Core Stage Forward Skirt Umbilical
CSITU	=	Core Stage Intertank Umbilical
EDL	=	Engineering Development Laboratory
GN2	=	Gaseous Nitrogen
HAASP	=	Hydraulic Arms and Accessories Service Pressure
ICPSU	=	Interim Cryogenic Propulsive Stage Umbilical
KSC	=	Kennedy Space Center
LCS	=	Launch Control System
LCC	=	Launch Command and Control
ML	=	Mobile Launcher
NASA	=	National Aeronautics and Space Administration
NE	=	Engineering and Technology Directorate
OSMU	=	Orion Service Module Umbilical
PAO	=	Public Affairs Office
SLS	=	Space Launch System
TSMU	=	Tail Service Mast Umbilical
VM	=	Virtual Machine
VS	=	Vertical Stabilizer

I. Introduction

In order to check out and launch rockets at National Aeronautics and Space Administration (NASA) Kennedy Space Center (KSC), an abundance of hardware components and parts are used and grouped into multiple subsystems. A new command and control system was needed to support the Space Launch System (SLS) as it had been undergoing heavy development in the past several years and was expected to be fully working in 2018. As a remote application software developer, a series of tasks are required, including getting trained, design, development, integration, and verification on the software and hardware for the firing rooms. Remote displays use Java technology on a Linux platform while the display development consists of drag and drop. An abstraction layer called the Application Control

[1] Undergraduate Student, Dept. of Aeronautics and Astronautics, The Ohio State University

Language (ACL) sits on top of the C++ language and allows non-software engineers to comprehend the source code for the individual subsystem, which could potentially be maintained for 30 years or more.

The initial phase of the project included training and demonstrating the skills and abilities required to create a remote control application and a remote display using the tools on the Linux Virtual Machine (VM). After the completion of the initial training and verification by a mentor, the Launch Accessories subsystem was assigned to a developer. The remote display developer had to follow the particular team's schedule and project goals. The software development process of designing and documenting the software, developing the suitable software, testing and documenting the unit, integrating the software with launch control system hardware and models, fixing and documenting bug fixes, writing and executing verification tests were carried out in sequence.

The duration of the internship did not permit the developer to complete a full software development life cycle. But it was crucial to complete and perform the work required for the particular phase of the specific subsystem team.

II. Objectives

The overall goal of the fall internship was to assist Launch Accessories subsystem team. In order to meet the goal and requirements, the following objectives were set and to be completed in sequence:

1. Review the Hitchhiker's Guide and obtain every IT account that is associated with being a remote developer.
2. Review and understand all the training documents on the wiki website as part of the training
3. Shadow one or more current subsystem remote software developers while attending subsystem meetings to perform software requirements and design tasks. The intent is to become familiar with Design, Development, Integration, and Verification phases of the life cycle.
4. Create an example remote display using the appropriate tools on the Linux VM. The requirements were to include a background image, at least one of each of the 4 different widgets, tie the window to any appropriate measurement of command that already exists in the database. The display must contain several text measurement widgets. One must be an analog/float measurement formatted to show at least 2 decimal places. A numeric measurement whose name has an "F" in the second to last character, which represents a float, while "I" is used to represent an integer. Another text measurement widget must be an analog/float measurement formatted to show only integer value. Run the display using the Display Test Driver and configure the measurements to verify the widgets display. A screen shot of the example display running with data in the widgets was required to be taken and sent to the mentor for approval[2].
5. Create an example remote control application using the appropriate tools on the Linux VM. This particular example application should be looking at the values from two separate discrete measurements. If the measurements values were both true then the application writes the string "We are GO!" to a Pseudo measurement. This application should use appropriate measurements that already exist in the database. A screen shot of the example in the development environment showing the source code and showing the successful compiling in the compiler output was required to be taken and sent to the mentor for approval[2].
6. After obtaining approval from the mentor of the skills demonstrated, the Launch Accessories subsystem teams were assigned. All the software development process including designing and documenting the software, developing the suitable software, testing and documenting the unit, integrating the software with hardware and models, fixing and documenting bug fixes, writing and executing verification tests will be conducted with the needs of different subsystem team depending on the progress.

III. Setup

The design and development part of the process was completed in the offices of the KSC Operation Support Building II with the support of mentors and experienced launch command and control personnel. Windows computers and appropriate software were installed with instructions. The software testing and integration part of the project were carried out in the firing rooms of the KSC Launch Command and Control (LCC) building with appropriate setup and support provided by the firing room personnel. Hardware testing and integration were conducted in the Launch Equipment Testing Facility (LETF) of KSC where simulated rocket force and motion were provided and the umbilical arms were installed on a lower platform instead of directly on the Mobile Launcher (ML). The hardware testing and integration, which we called LETF testing, provides a safer and easier testing environment prior to the testing on the ML.

IV. Results

A. Skills Demonstration

After completing the first three objectives a sample remote display was created as shown in Figure 1 based on the requirements. This display included measurements of different data types, command, indicator, and button linked to another display. The measurements had limits and bounds that were represented by different colors, the units were tied to the measurements. Also the display included a random background picture.

The coding part of the skills demonstration was coded in NetBeans. By successfully compiling the code, the developer has demonstrated and completed the code skills demonstration and was ready to be assigned to a subsystem.

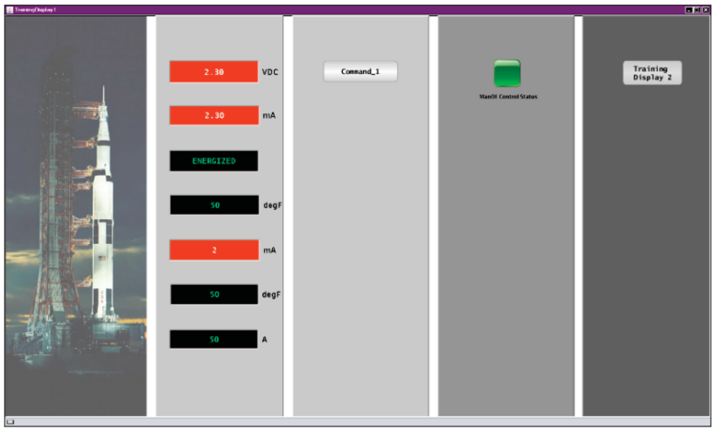


Figure 1. Display Skills Demonstration

B. Launch Accessories Subsystem

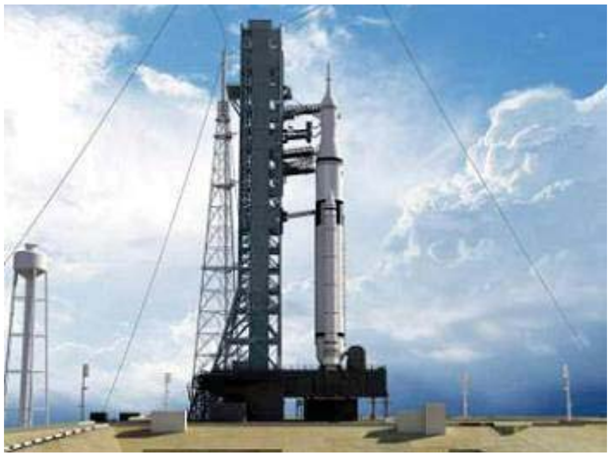


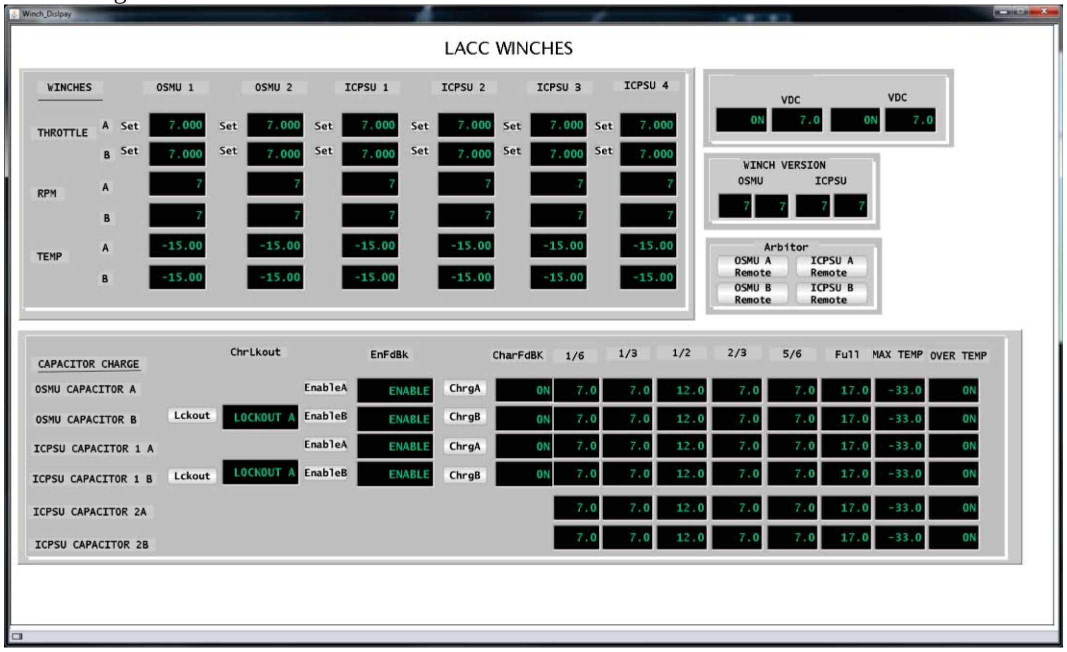
Figure 2. ML and SLS. Image source: NASAspaceflight[3]

The Launch Accessories Subsystem is made up of fourteen arms and umbilical subsystems for which there are ten software projects assigned to the developer. As shown in Figure 2, the SLS will be placed on the ML. The umbilical arms, which provides crew access, cryogenic, and fuels for SLS, will be installed on the ML and linked to the SLS. The arms also monitor the health status and stabilize the vehicle[3]. The arms consists of Crew Access Arm (CAA), Orion Service Module Umbilical (OSMU), Interim Cryogenic Propulsive Stage Umbilical (ICPSU), Core Stage Forward Skirt Umbilical (CSFSU), Core Stage Intertank Umbilical (CSITU), Vehicle Stabilizer (VS), Tail Service Mast Umbilical (TSMU), and Aft Skirt Purge Umbilical (ASPU).

The subsystem had accessories including Hydraulic Arms and Accessories Service Pressure (HAASP), Winch for OSMU and ICPSU, and Gaseous Nitrogen (GN2) Pressure. The objectives of the remote software developers were to make the arms extend to and retract from the vehicle by controlling the valves and hydraulic supply on the remote display. The arm movement is directly tied to the amount of hydraulic pressure supplied. All the umbilical arms are T-0 arms except for CAA, which means automated retract sequences are executed when the rocket launches at T-0. There are latchback mechanisms designed for most of the umbilical arms to prevent damaging the vehicle and the arms themselves. Redundancy plays an important factor for software developers in order to carry out a safe and successful mission.

1. *Winch Display*

The author entered the software-testing phase of the project and was given task to test and modify the current Winch Display. Figure 3 presents the Winch Display after being modified and reorganized. Level 1 testing, which is the most basic and first level display testing that has no hardware components or logic connected to the display, was conducted in the firing room to verify the response of each indicator after forcing the values in a simulated model. The display was tested through Simulated Control by forcing values to the feedback global identifiers and observing appropriate changes on the display. Also, commands were issued through Simulated Control to see any response. Level 1 testing can be passed only if all the global identifiers on the display are verified to be working properly. Prior to the level 1 testing, all the global identifiers were checked and verified between the user generated spreadsheet and the identifier bank in the test configuration.



LACC WINCHES

WINCHES		OSMU 1	OSMU 2	ICPSU 1	ICPSU 2	ICPSU 3	ICPSU 4
THROTTLE	A	Set 7.000	Set 7.000	Set 7.000	Set 7.000	Set 7.000	Set 7.000
	B	Set 7.000	Set 7.000	Set 7.000	Set 7.000	Set 7.000	Set 7.000
RPM	A	7	7	7	7	7	7
	B	7	7	7	7	7	7
TEMP	A	-15.00	-15.00	-15.00	-15.00	-15.00	-15.00
	B	-15.00	-15.00	-15.00	-15.00	-15.00	-15.00

	VDC		VDC
ON	7.0	ON	7.0

WINCH VERSION

OSMU		ICPSU	
7	7	7	7

Arbitor

OSMU A Remote	ICPSU A Remote
OSMU B Remote	ICPSU B Remote

CAPACITOR CHARGE		ChrLkout		EnFdBk		CharFdBK	1/6	1/3	1/2	2/3	5/6	Full	MAX TEMP	OVER TEMP
OSMU CAPACITOR A			EnableA	ENABLE	ChrgA	ON	7.0	7.0	12.0	7.0	7.0	17.0	-33.0	ON
OSMU CAPACITOR B	Lckout	LOCKOUT A	EnableB	ENABLE	ChrgB	ON	7.0	7.0	12.0	7.0	7.0	17.0	-33.0	ON
ICPSU CAPACITOR 1 A			EnableA	ENABLE	ChrgA	ON	7.0	7.0	12.0	7.0	7.0	17.0	-33.0	ON
ICPSU CAPACITOR 1 B	Lckout	LOCKOUT A	EnableB	ENABLE	ChrgB	ON	7.0	7.0	12.0	7.0	7.0	17.0	-33.0	ON
ICPSU CAPACITOR 2A							7.0	7.0	12.0	7.0	7.0	17.0	-33.0	ON
ICPSU CAPACITOR 2B							7.0	7.0	12.0	7.0	7.0	17.0	-33.0	ON

Figure 3. Winch Remote Display

2. *ICPSU Display*

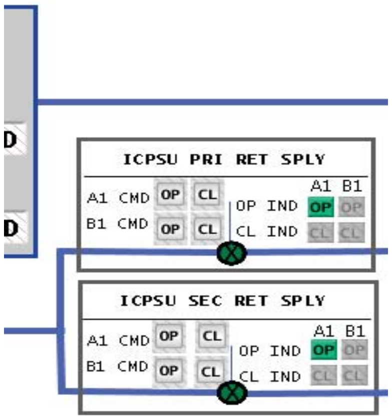


Figure 5. Old ICPSU Display

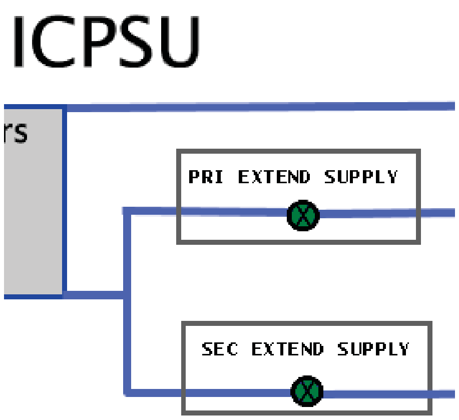


Figure 5. New ICPSU Display

The previous version of the ICPSP remote display contains everything needed to operate ICPSU. It is important to note that the console users would not be in favor of a complicated display. There was excessive amount of the primary and secondary controls and indicators as shown in Figure 4, which may confuse any first time operator. Design and discussion meetings were attended and the customer demanded simplification and pop-up style display.

The new ICPSU Remote Display simplified and deleted most of the valve controls and indicators as shown in Figure 5. Only value open or closed indicators were left on the display. If the operator wishes to control or see the primary and secondary indicators of a particular valve, then a pop-up display can be opened by clicking on the circle valve symbol. For example the ICPSU HAASP Valve information can be opened by clicking the ICPSU HAASP valve button and a pop-up display will appear as show in Figure 6. Extend supply, extend return, retract supply, and retract return have been grouped into four pop-up displays. Figure 7 shows the ICPSU extend supply valve display. The latch back valves also had their own display. This new display method simplified the main display in order for console users to monitor multiple subsystems with the least amount of manpower.

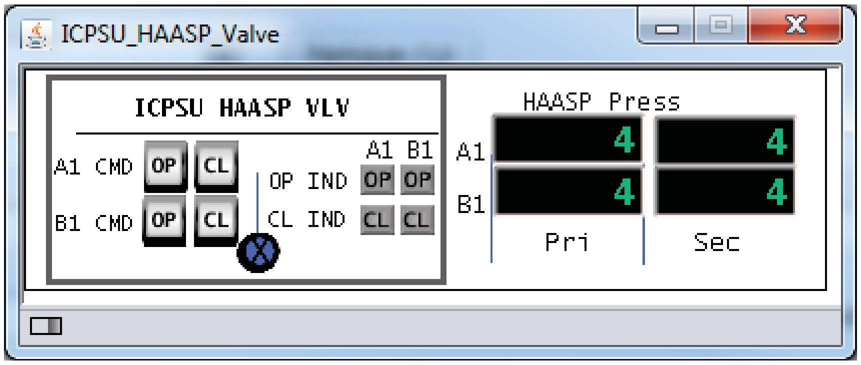


Figure 6. ICPSU HAASP Valve Display

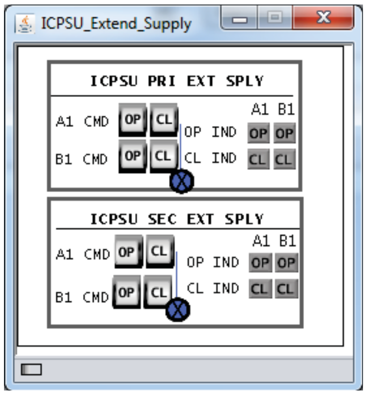


Figure 7. ICPSU Extend Supply Display

3. *Power Bus and other Displays*

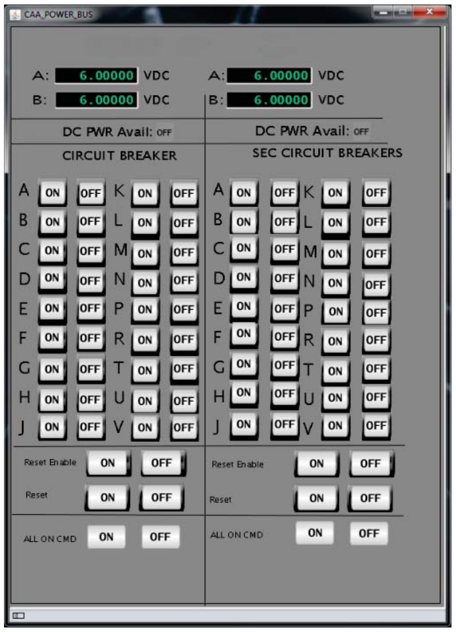


Figure 8. CAA Power Bus

Power Bus Displays were created for CAA, ICPSU, CFSU, and CSITU. The displays were copied and identical in design. The command and indicator identifiers were different depending on the umbilical arm, and the RIO number was different for each arm.

Figure 8 shows the CAA Power Bus as an example. The rest of the Power Bus displays were originated from the CAA Power Bus. As shown in the figure, Power Bus consists of circuit breakers for A and B sides. In order for the umbilical arm system to work, all of the breakers must be turned on. This can be done by issuing individual commands to individual breakers until the power available indicator turn on, or All On command can be send to flip all the breakers on.

Other displays including CSFSU and CSITU were designed based on the ICPSU display. CSFSU and CSITU were almost identical in terms of arm movement and valve control.

C. Additional Displays and Data Conversion Fusion

Measurement data conversion display and ACL scripts were created to support a Kickstart project for public-friendly remote displays. In Figure 9, the Public Affairs Office (PAO) fusion display was created to convert a global identifier value to another global identifier value with different units. These conversion included unit conversion in altitude, velocity, and acceleration. ACL scripts and rules were created and executed behind the scene every time the input global identifier changes.

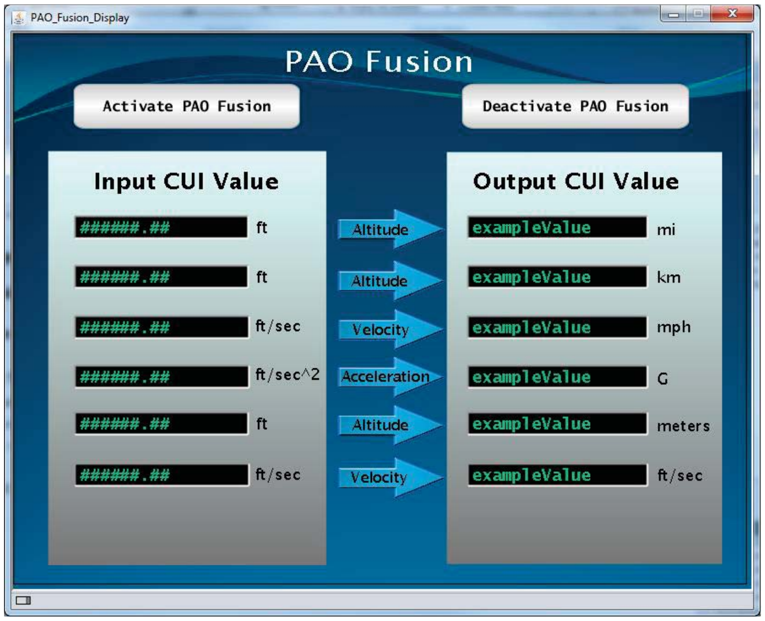


Figure 9. PAO Fusion Display

There were two command buttons on the top of the display served as active and deactivate purpose. When active was pressed, the fusion ACL scripts were executed immediately to convert the input global identifier value into another unit based on the requirements, saved the value to another global identifier, and then displayed onto the output value panel. The deactivate PAO Fusion button terminates the scripts and stops all conversion when pressed.

V. Conclusion

As expected, the duration of the internship did not permit the developer to complete a full software development life cycle, but it was essential to know that documentation is strongly recommended for a single phase project. As for deliverables, a modified Winch Display was submitted and integrated to help resolve the current testing issues. The rest of the displays including ICPSU, CSFSU, CSITU, and Power Bus Displays for CAA, ICPSU, CSFSU, and CSITU were created by the developer and are ready in the queue to be promoted for later integration into the system.

Electrical and 90% software requirement document tabletop meetings were attended to discuss the status of the design requirements, and to raise questions and concerns. Also, informal meetings were held with subsystem team engineers to better understand and simplify the umbilical arm displays. An abundance of time was spent in the firing room for level 1 testing on the Winch Display for OSMU and ICPSU. The basic operation procedure for the firing room integrated command and control system was taught by the firing room experts to launch the displays and to use the tools. The remote developer had been working closely with the local developers in the Engineering Development Laboratory (EDL) to perfect the local Allen Bradley display, especially fixing issues on the Local Winch Display in order for OSMU LETF Testing to be carried out on time.

Also the PAO Fusion display was created for the firing room integrated command and control system set. Component build request was sent based on the software development procedure and approved later on. The display was later verified visually on an integrated command and control system set in the firing room.

VI. Acknowledgments

The author would like to thank and acknowledge Kurt Leucht and Steve Camick for the support and mentoring work to this project. Thanks to Dave Stewart for the additional help. Also special thanks to KSC Engineering and Technology Directorate and the Education Office, without whom this project would not have been possible.

VII. References

[2]Hitchhiker's Guide. (2014). [online] [Accessed 10 Sep. 2014].

[3]"NASA Outlines SLS Mobile Launcher Umbilical Plans." NASASpaceFlight.com. N.p., n.d. Web. 04 Nov. 2014.

[4]nasa-ice, (2014). LCS Integrated Launch Operations Application Software. [online] Available at: https://nasa-ice.nasa.gov/confluence/display/GOLCSILOA/ILOA+Training [Accessed 10 Sep. 2014].